Chapter5: Intro to Adversarial Search

Part I

Outline

Adversarial Search
Optimal decisions
Minimax
α-β pruning
Case study: Deep Blue
UCT and Go

Adversarial Reasoning: Games

Mathematical Game Theory

Branch of economics that views any multi-agent environment as a game, provided that the impact of each agent on the others is "significant", regardless of whether the agents are cooperative or competitive.

First step:

- **Deterministic**
- **Turn taking**
- **2-player**
- **Zero-sum game of perfect information (fully observable)**

"my win is your loss" and vice versa; utility of final states

Game Playing vs. Search

Multi-agent game vs. single-agent search problem

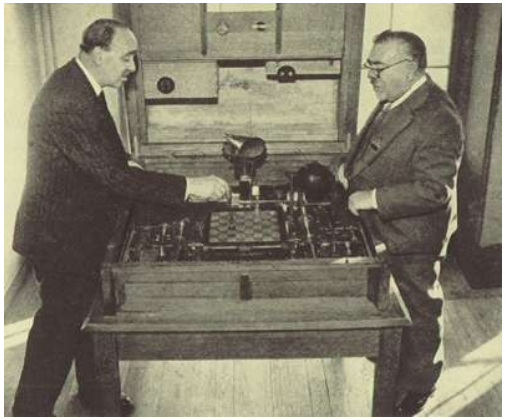
"Unpredictable" opponent need a strategy: specifies a move
for each possible opponent reply.
E.g with “huge” lookup table.

Time limits unlikely to find optimal response, must approximate

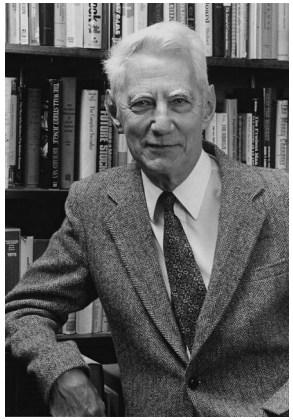
Rich history of game playing in AI, in particular in the area of chess.

Both Turing and Shannon viewed chess as an important challenge for
machine intelligence because playing chess appears to require some
level of intelligence.

A Brief History of Computer Chess



1912



1950
s

1970
s

199
7

Toda

Human-computer hybrid most exciting new level of play. Comput
as smart assistants are becoming accepted.
Area referred to as "Assisted Cognition."

Another example: mind-reading binoculars for 10 km vision.

Why is Game-Playing a Challenge for AI?

Competent game playing is a mark of some aspects of "intelligence"

- **Requires planning, reasoning and learning**

Proxy for real-world decision making problems

- **Easy to represent states & define rules**
- **Obtaining good performance is hard**

"Adversary" can be nature

PSPACE-complete (or worse)

- **Computationally equivalent to hardware debugging, formal verification, logistics planning**
- **PSPACE believed to be harder than NP.**

Traditional Board Games

Finite

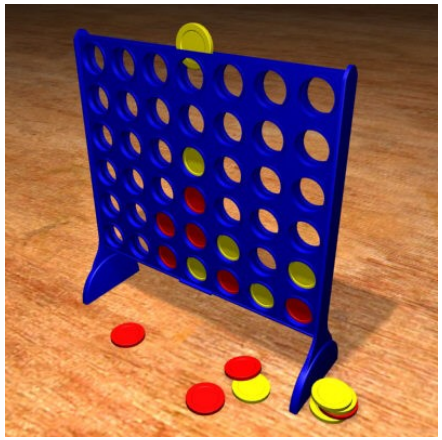
Two-player

Zero-sum

Deterministic

Perfect Informa

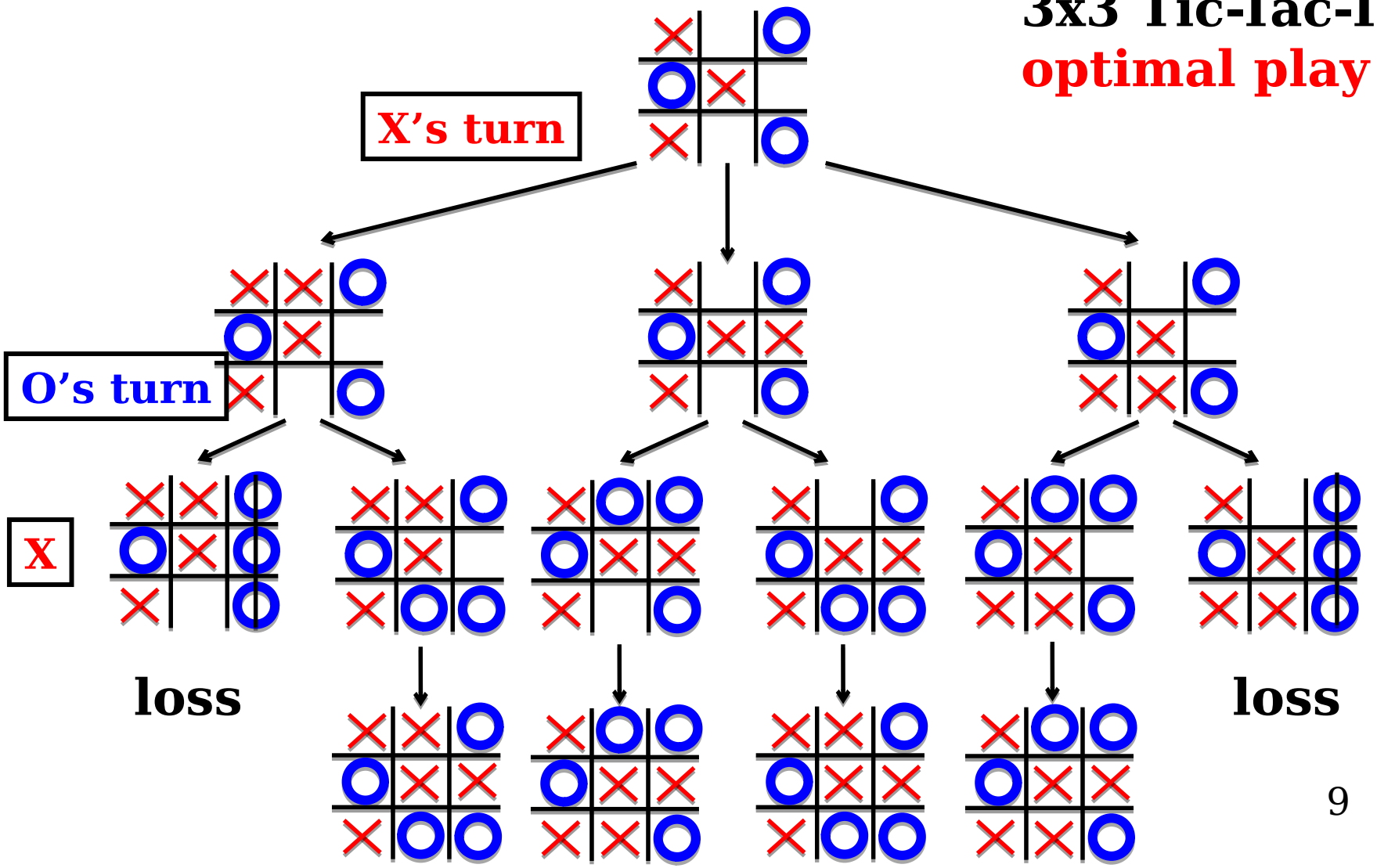
Sequential



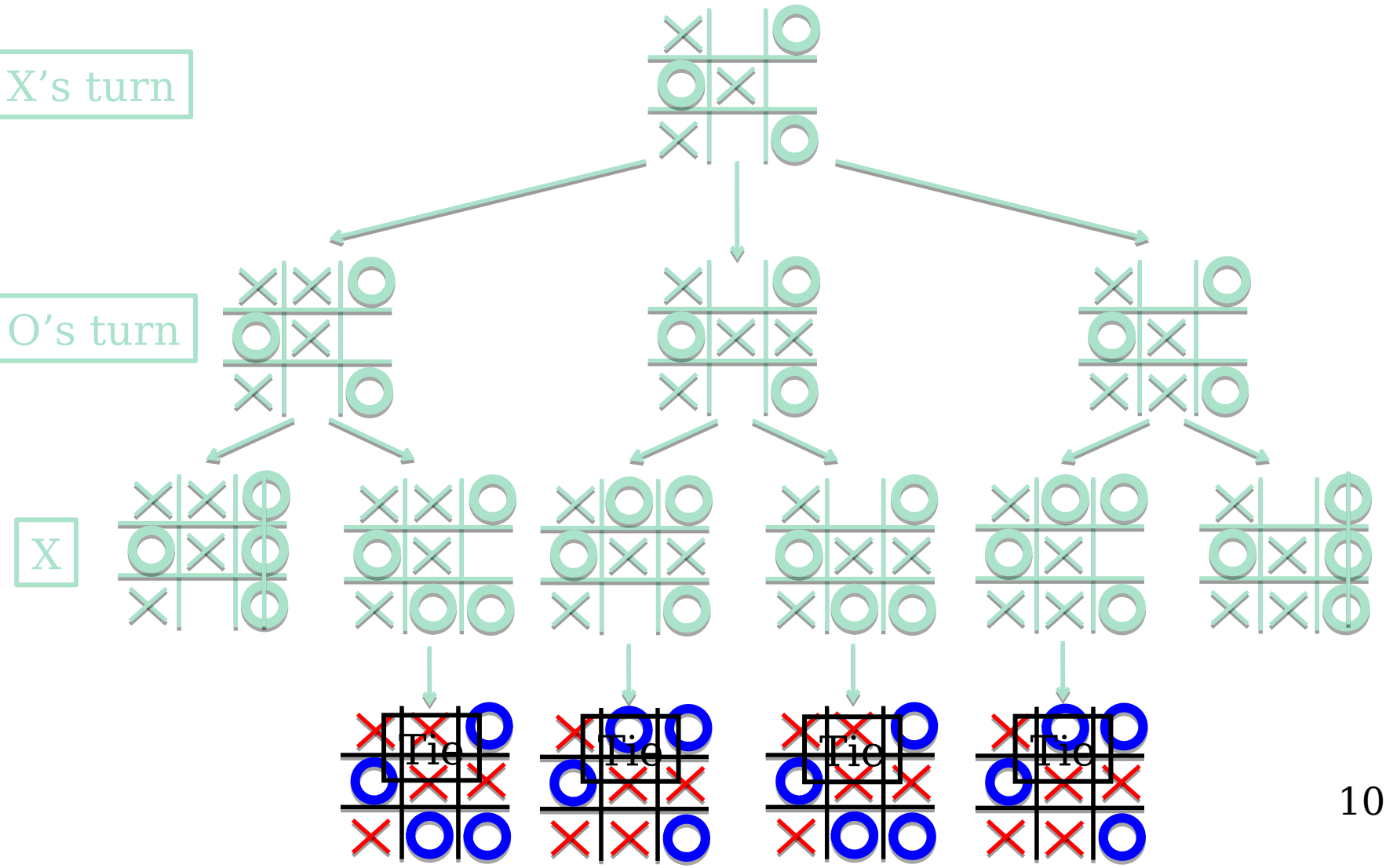
Tic-tac-toe (or Noughts and crosses, Xs and Os)

Key Idea: Look Ahead

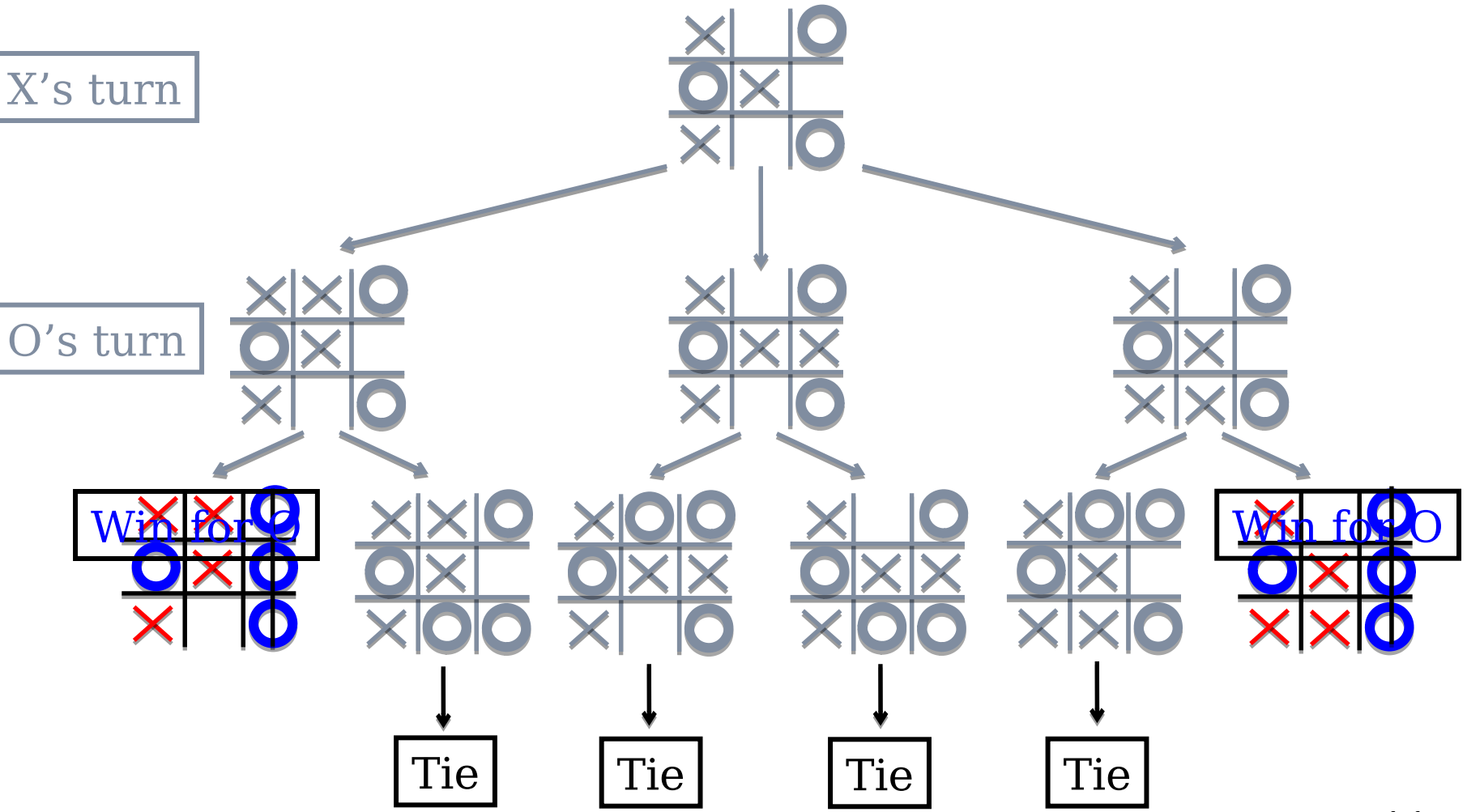
We start 3 moves per player in:



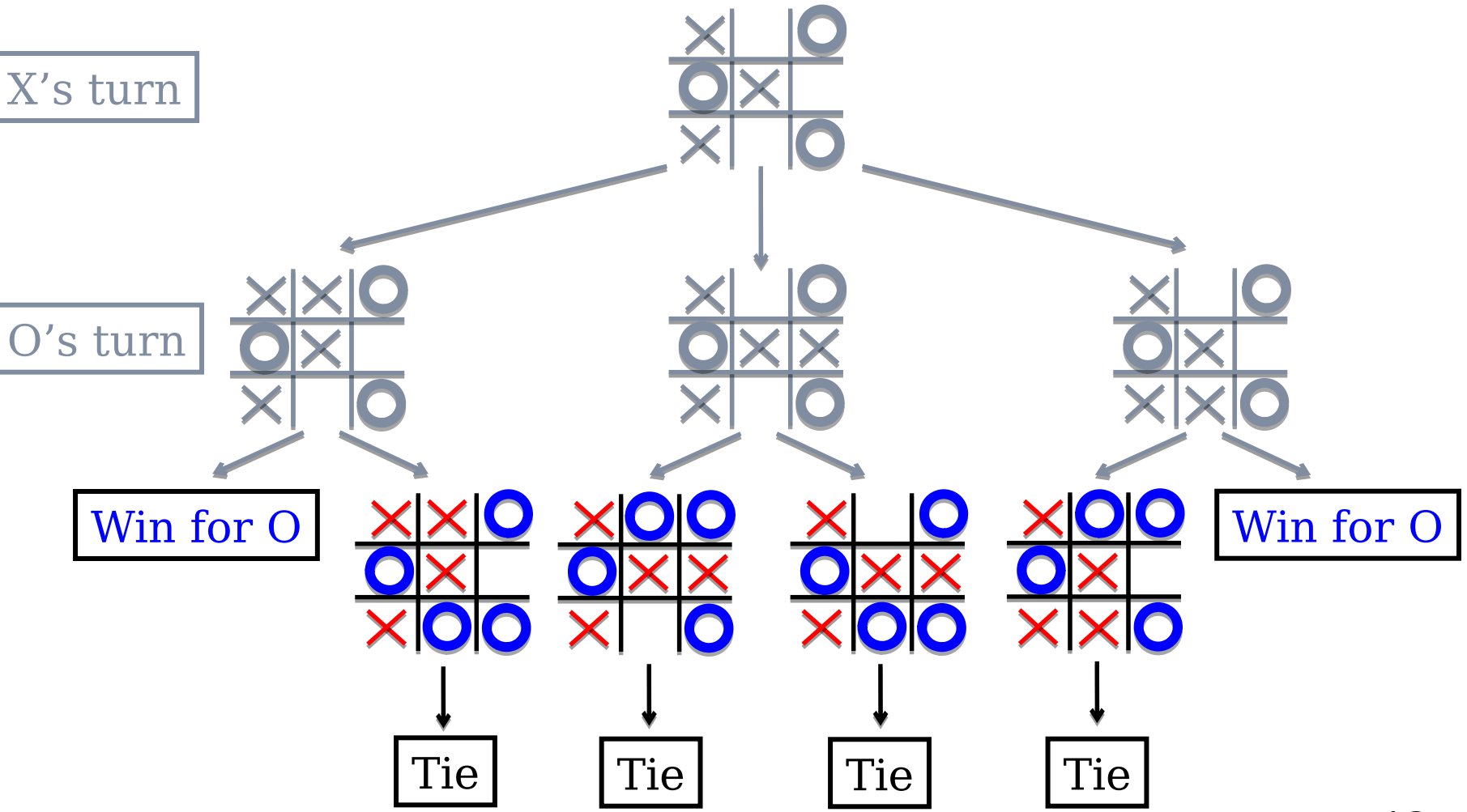
Look-ahead based Tic-Tac-Toe



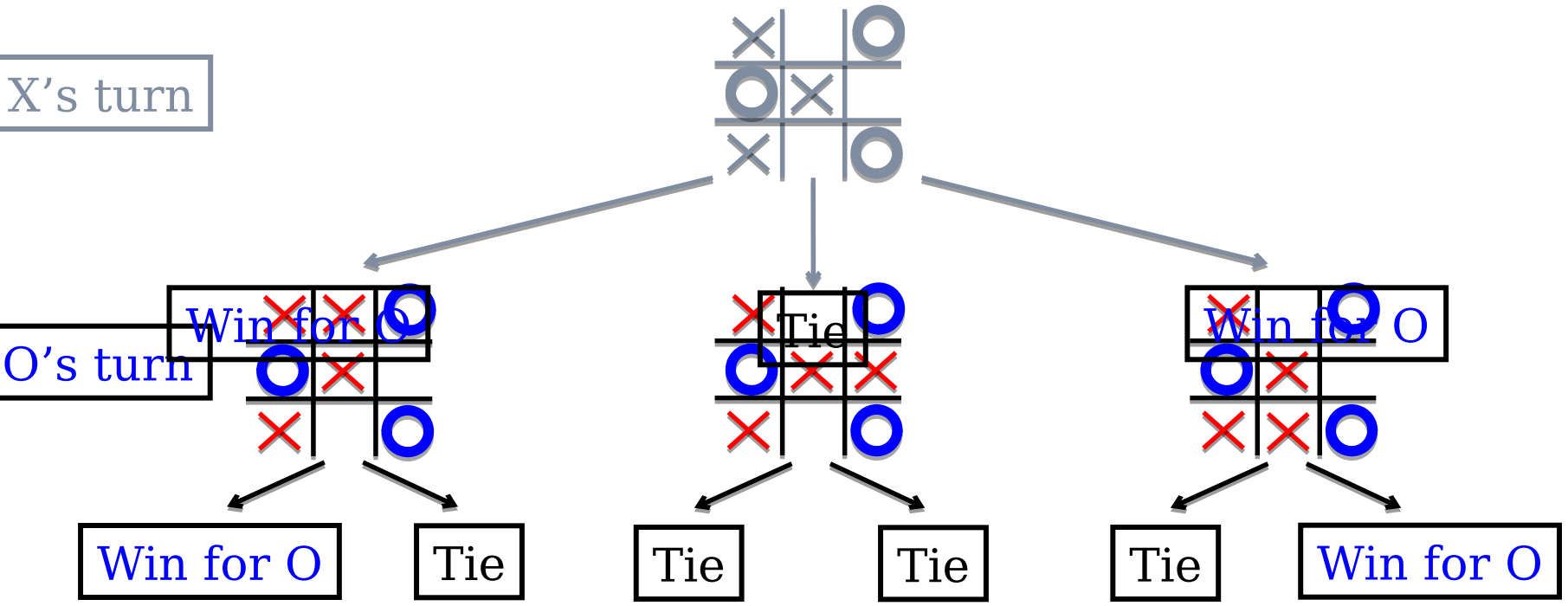
Look-ahead based Tic-Tac-Toe



Look-ahead based Tic-Tac-Toe



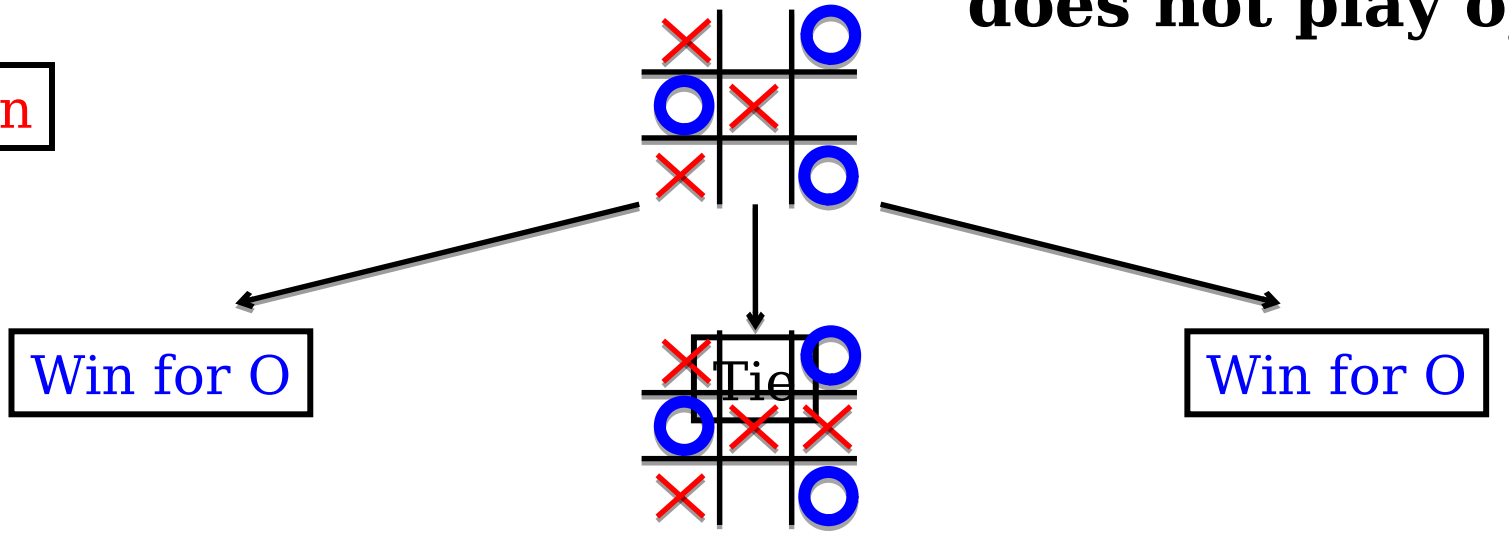
Look-ahead based Tic-Tac-Toe



h board in game tree gets unique
ne tree value (utility; -1/0/+1)
er optimal rational play.
nvince yourself.)

E.g. 0 for top board

What if our opponent does not play optimal



Approach: Look first at bottom tree. Label bottom-most boards.

Then label boards one level up, according result of best possible move.

... and so on. Moving up layer by layer.

Termed the Minimax Algorithm

- **Implemented as a depth-first search**

Aside: Game tree learning

Can (in principle) store all board values in large table. 3^9 = 19.683

for tic-tac-toe. (each cell could be in 3 states (X ,O, Empty), 9 ?

Can use table to try to train classifier to predict “win”, “loss”, or

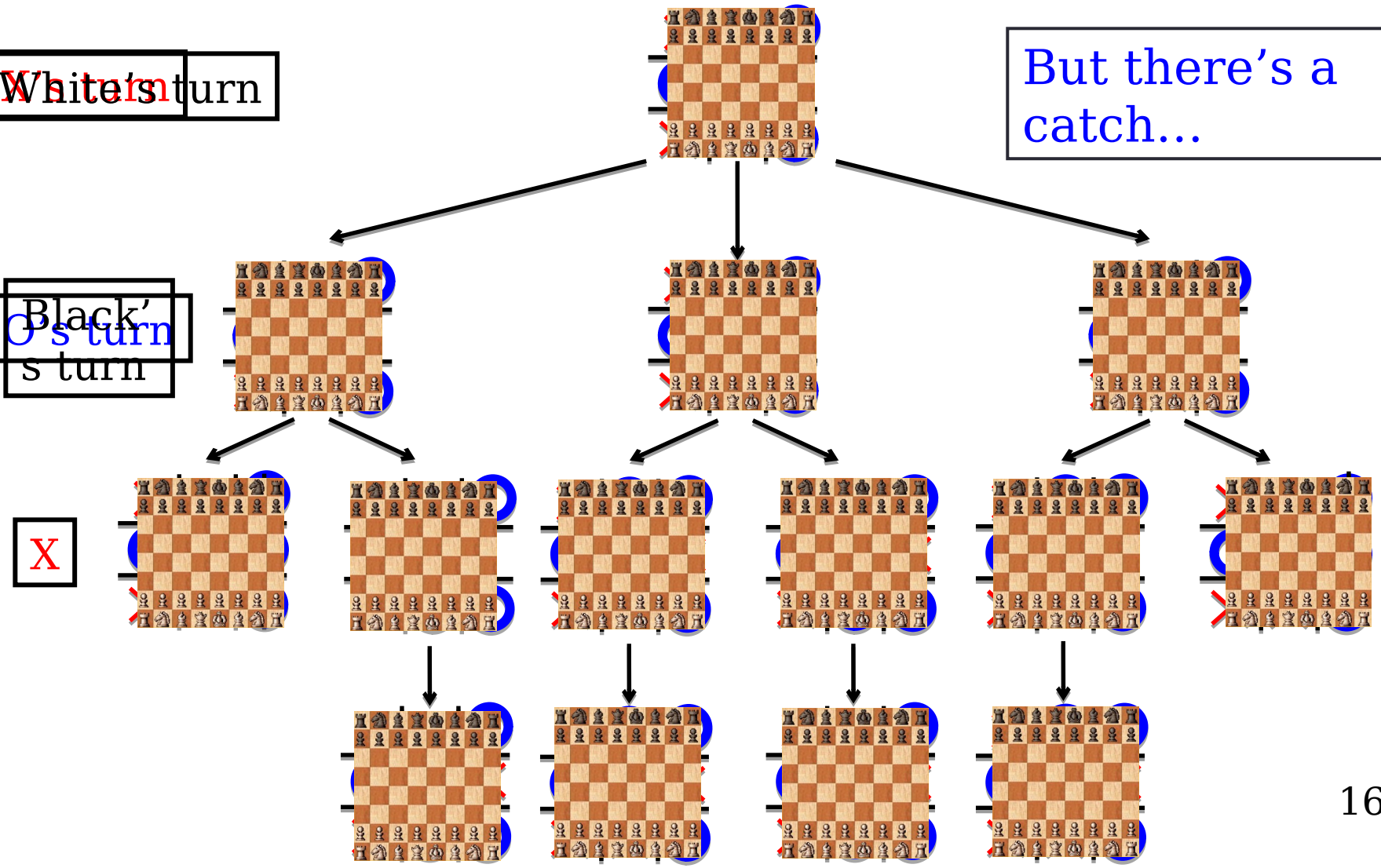
“draw.”

Issue: For real games, one can only look at tiny, tiny fragment of

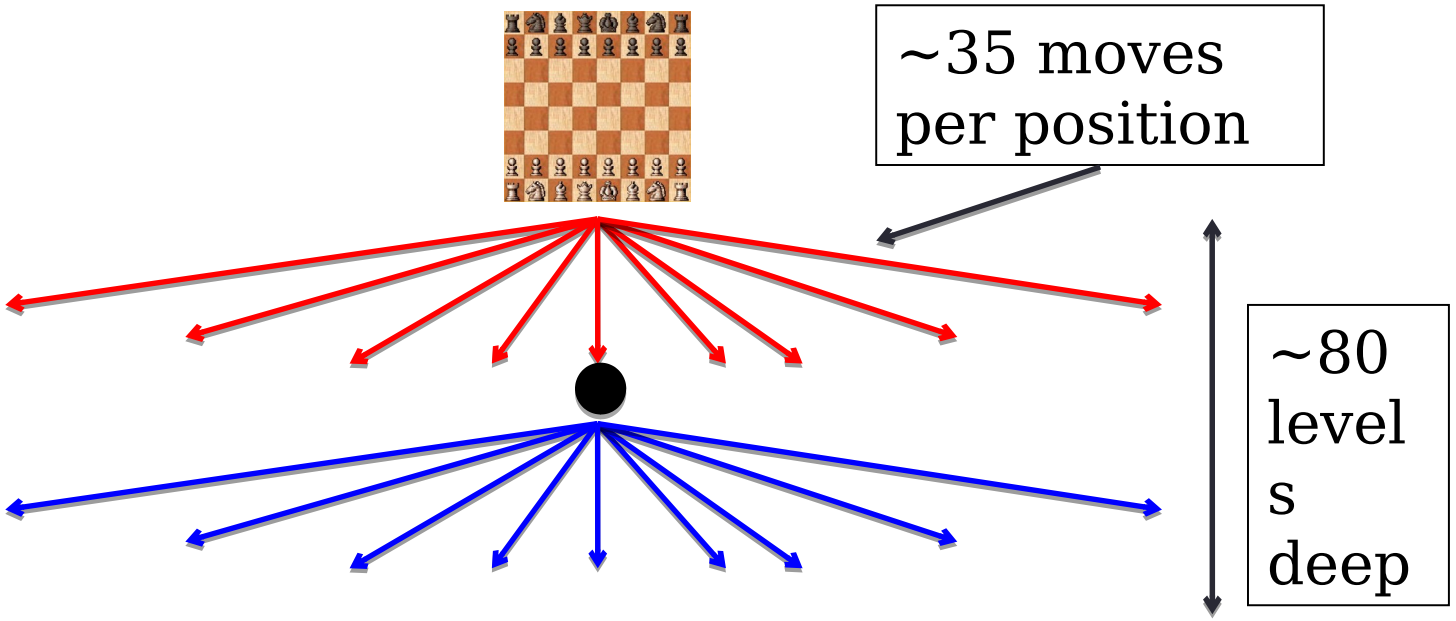
table.

Reinforcement learning builds on this idea.

Look-ahead based Chess



How big is this tree?



Approx. 10^120 > Number of atoms in the observable universe (10^80)

We can really only search a **tiny, miniscule faction** of this tree!

What's the work-around?

Don't search to the very end

- **Go down 10-12 levels (still deeper than most humans)**
- **But now what?**
- **Compute *an estimate of the position's value***

c function is typically designed by a
rt

Consider a game tree with leaf utilities (final boards) +1 / 0 / -1 (or +inf / 0 –inf). What are the utilities of intermediate boards in the game tree?

+1 / 0 / -1
(or +inf / 0 / -inf)

The board heuristics is trying to *estimate* these values from a quick calculation on the board. Eg, considering material won/loss on chess board or regions captures in GO. Heuristic value of e.g. +0.9, suggests true value may be +1.

What is a problem for the board heuristics (or evaluation functions) at the beginning of the game?

(Consider a heuristics that looks at lost and captured pieces.)

hat will the heuristic values be near the top?

Close to 0! Not much has happened yet....

er issue: children of any node are mostly quite similar. es almost identical heuristic board values. Little or no ormation about the right move.

Solution: Look ahead. I.e., build search tree several levels deep (hopefully 10 or more levels). Boards at bottom of tree more diverse. *Use minimax search to determine value of starting board, assuming optimal play for both players.*

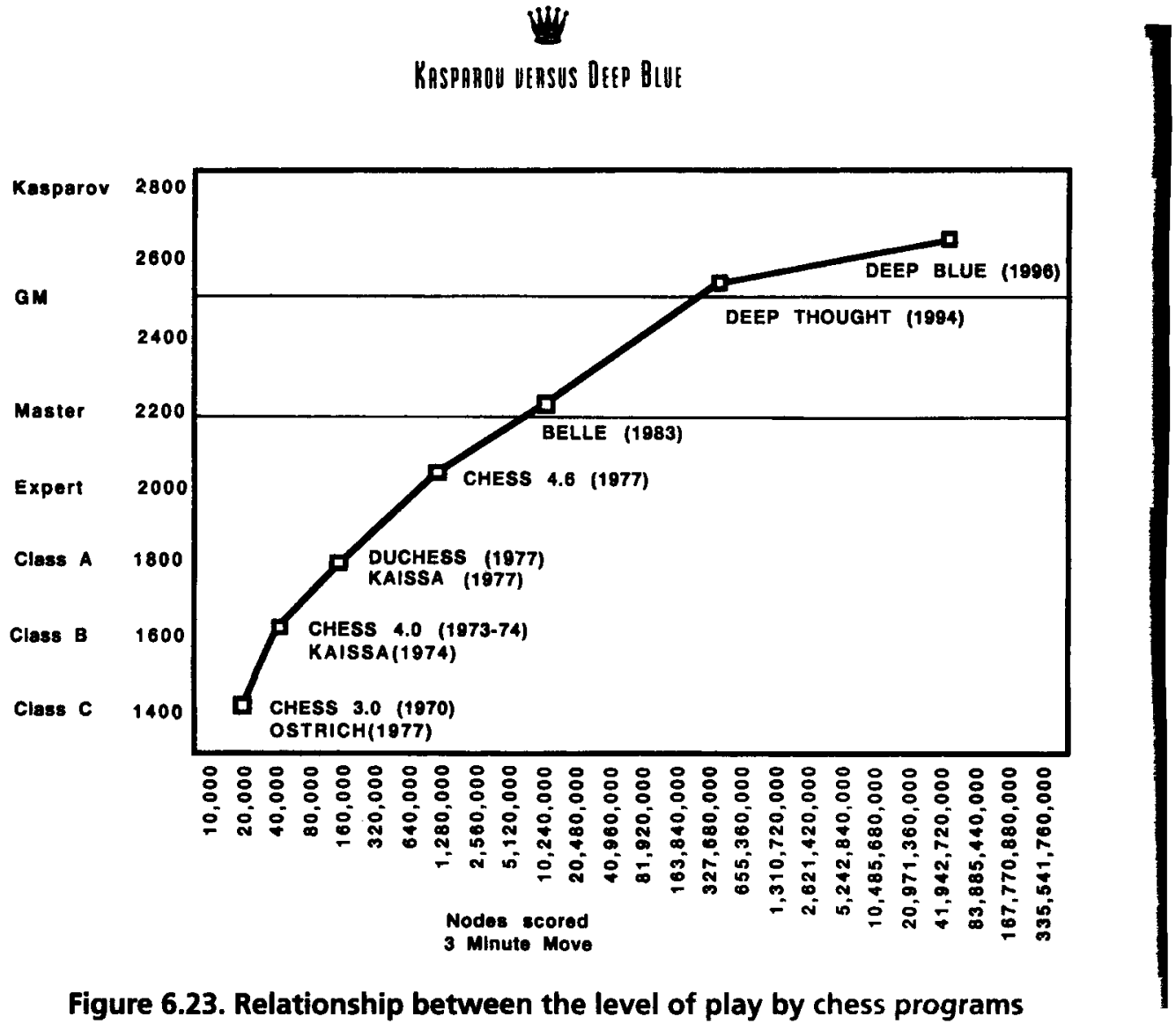


Figure 6.23. Relationship between the level of play by chess programs

ll deeper search give stronger play? Always? And why?

y counterintuitive: there are "artificial games" where searching eper leads to worse play! (Nau and Pearl 1980) Not in natural ga me tree anomaly.

Heuristic board eval value is sometimes informally referred to as the "chance of winning" from that position.

That's a bit odd, because in a deterministic game with perfect information and optimal play, there is no "chance" at all! Each board has a fixed utility: -1, 0, or +1 (a loss, draw, or a win). (result from game theory)

What if board eval gives true board utility? How much search is needed to make a move?

Still, "chance of winning" is an informally useful notion. But no clear semantics to heuristic values.

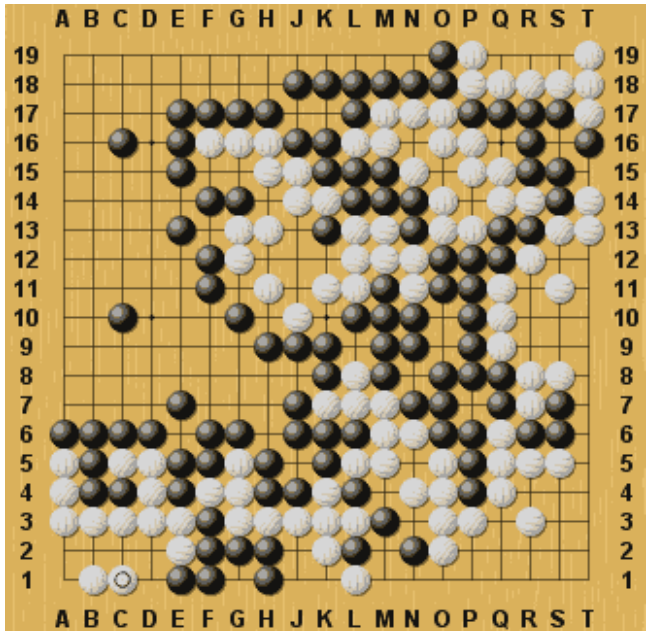
We'll see that using machine learning and "self play," we can get close for backgammon.

Limitations?

Two important factors for success:

- Deep look ahead
- Good heuristic function

Are there games where this is not feasible?



Limitations?

Two important factors for success:

- ~~Deep look ahead~~
- Good heuristic function

Are there games where this is not feasible?

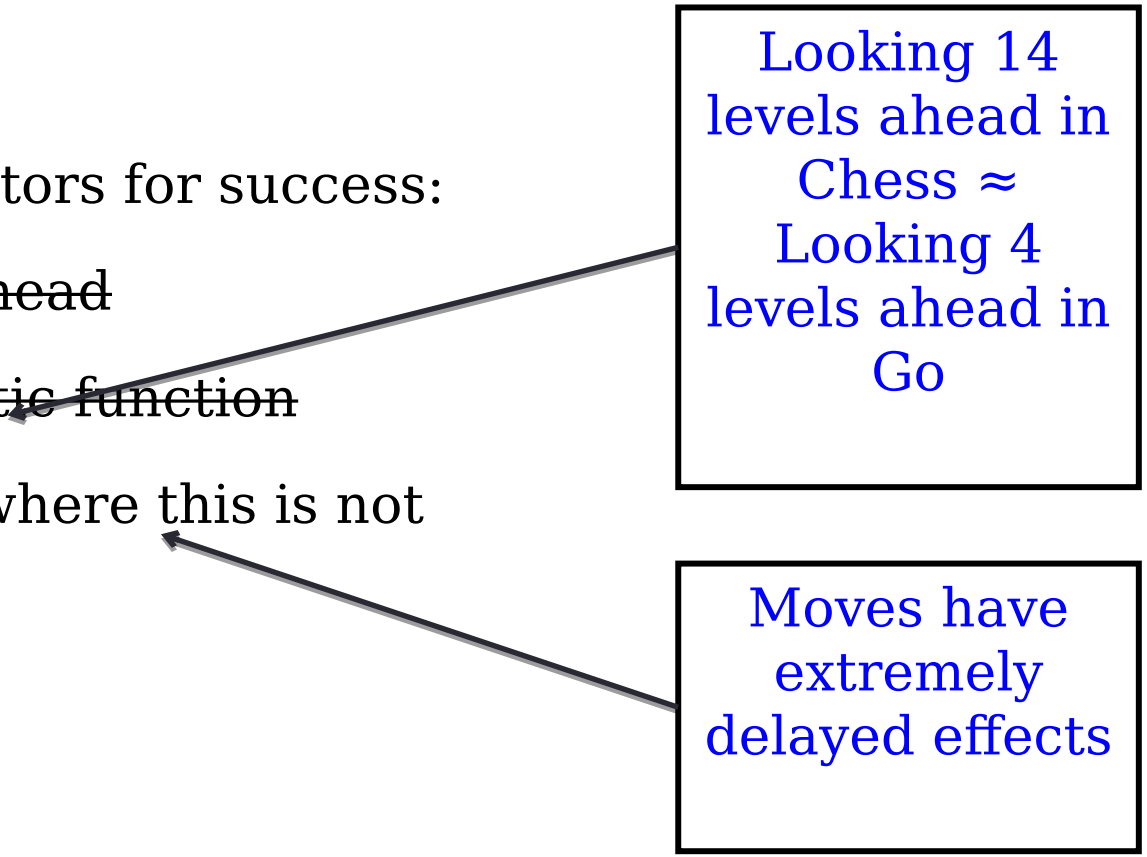
Looking 14 levels ahead in Chess ≈ Looking 4 levels ahead in Go

Limitations?

Two important factors for success:

- ~~Deep look ahead~~
- ~~Good heuristic function~~

Are there games where this is not feasible?



Limitations?

Two important factors for success:

- ~~Deep look ahead~~
- ~~Good heuristic function~~

Are there games where this is not feasible?

Looking 14 levels ahead in Chess ≈ Looking 4 levels ahead in Go

Moves have extremely delayed effects

Minimax players for GO were very weak until 2007…but now play at master level

Limitations?

Two important factors for success:

- ~~Deep look ahead~~
- ~~Good heuristic function~~

Are there games where this is not feasible?

Looking 14 levels ahead in Chess ≈ Looking 4 levels ahead in Go

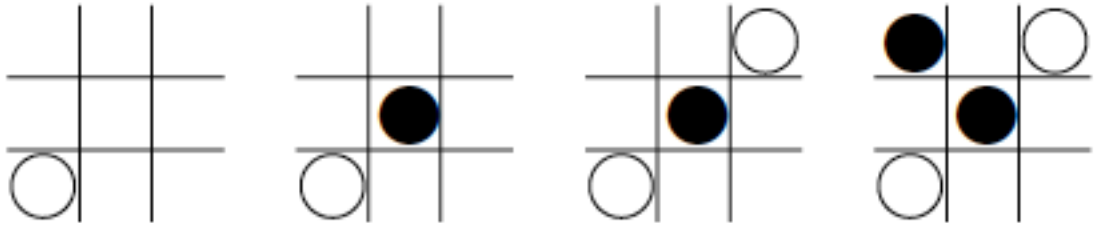
Moves have extremely delayed effects

New sampling based search method:
Upper Confidence bounds applied to Trees (UCT)

Well… Why not use a strategy / knowledge, as humans do?

Consider for Tic-Tac-Toe:

1. If there is a winning move, make it.
2. If the opponent can win at a square by his next move, play that move.
3. Taking the central square is more important than taking other squares.
4. Taking corner squares is more important than taking squares on the edges.



Oops!!

Consider Black uses the strategy…

So, although one can capture strategic knowledge of many games

in high-level rules (at least to some extent), in practice any

interesting game will revolve *precisely around the exceptions to*

those rules!

Issue has been studied for decades but research keeps coming back to

game tree search (or most recently, game tree sampling).

Currently only one exception: reinforcement learning for backgammon.

(discussed later)

A very strong board evaluation function was learned in self-play.

Formal definition of a game:

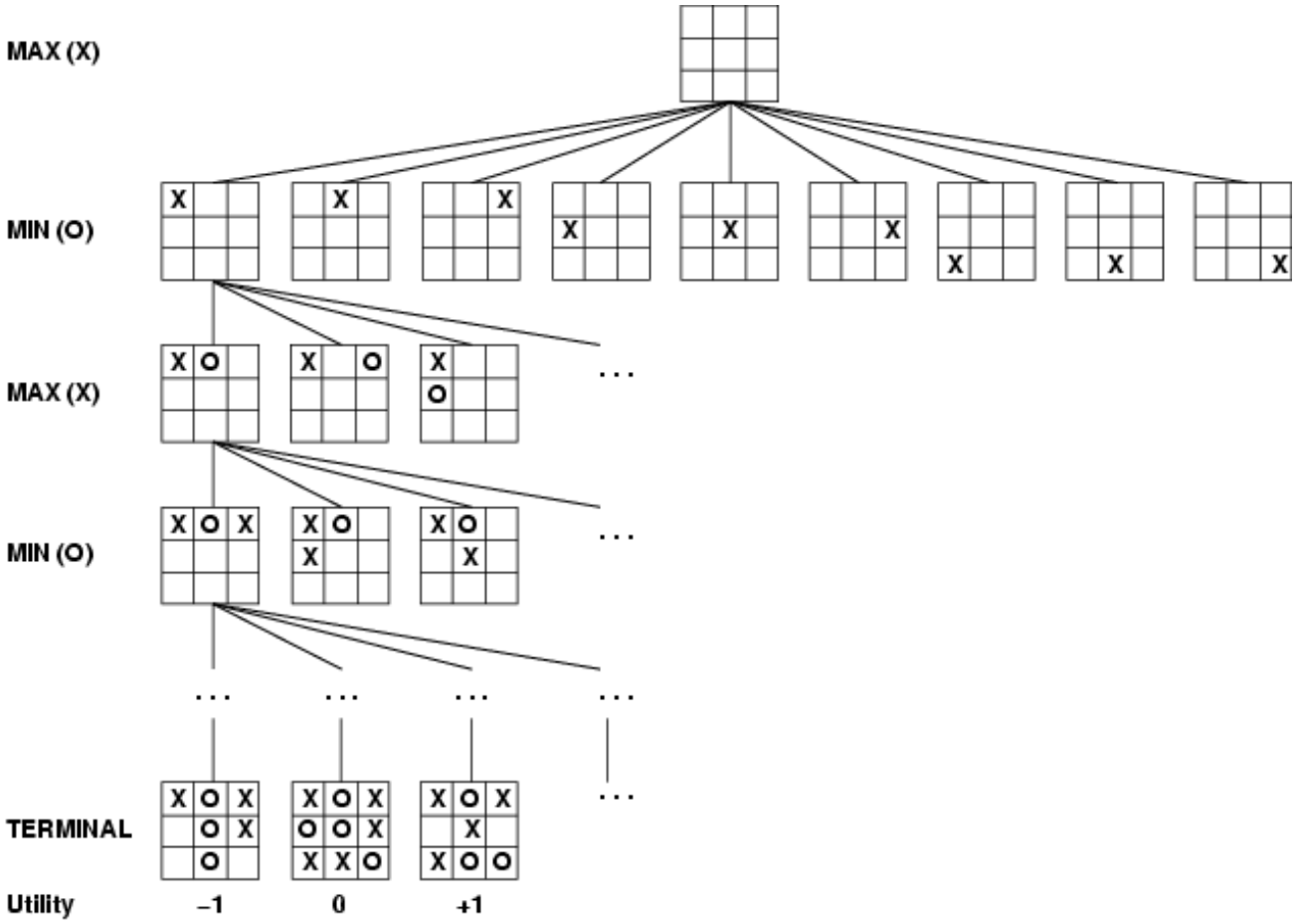
- **Initial state**
- **Successor function: returns list of *(move, state)* pairs**
- **Terminal test: determines when game over**

  Terminal states: states where game ends
- **Utility function (objective function or payoff function): gives numeric value for terminal states**

We will consider games with 2 players (**Max and Min**)

Max moves first.

Game Tree Example: Tic-Tac-Toe



Tree from Max's perspective

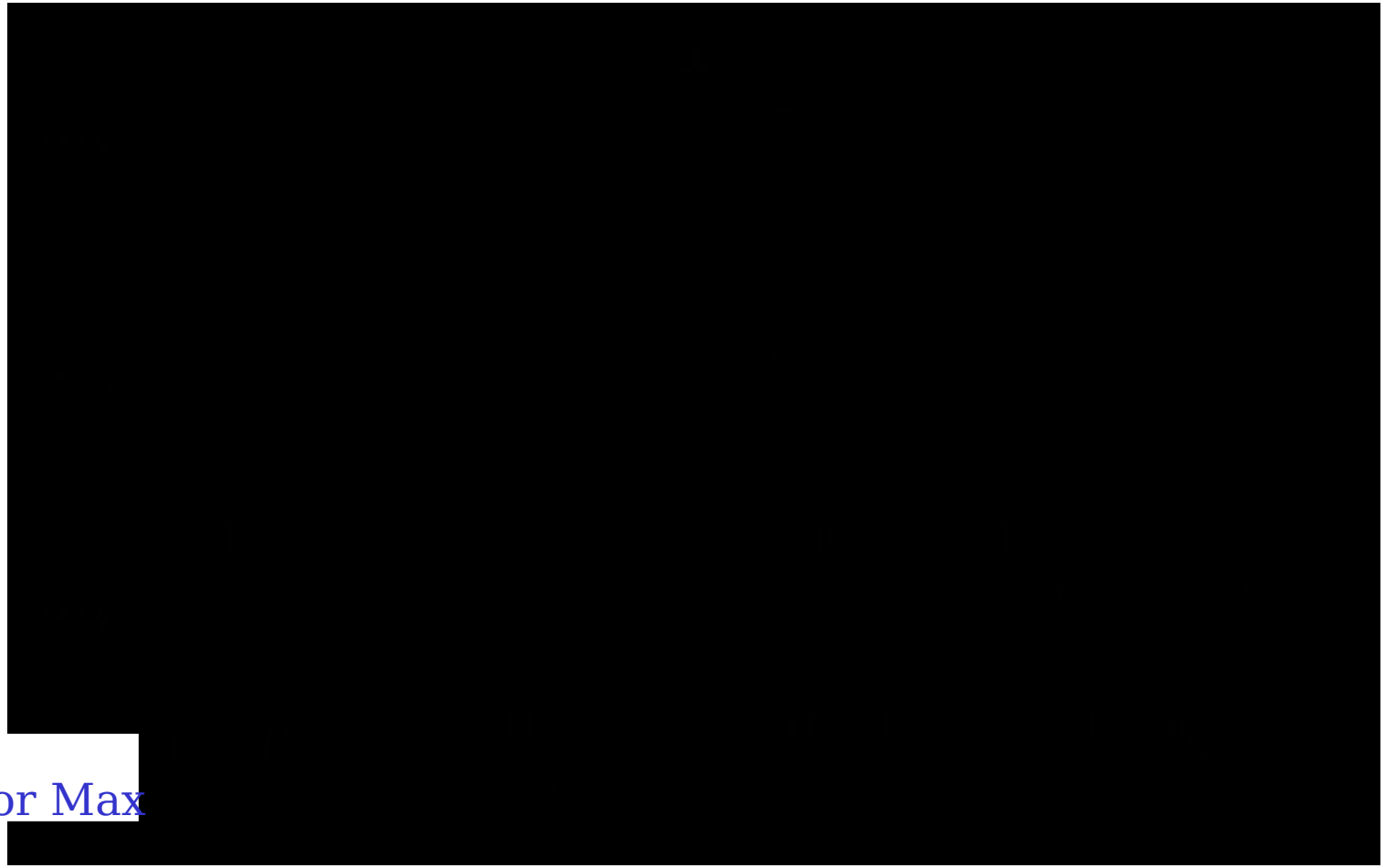
Minimax Algorithm

Minimax algorithm

- **Perfect play for deterministic, 2-player game**
- **Max tries to maximize its score**
- **Min tries to minimize Max's score (Min)**
- **Goal: Max to move to position with highest minimax value**

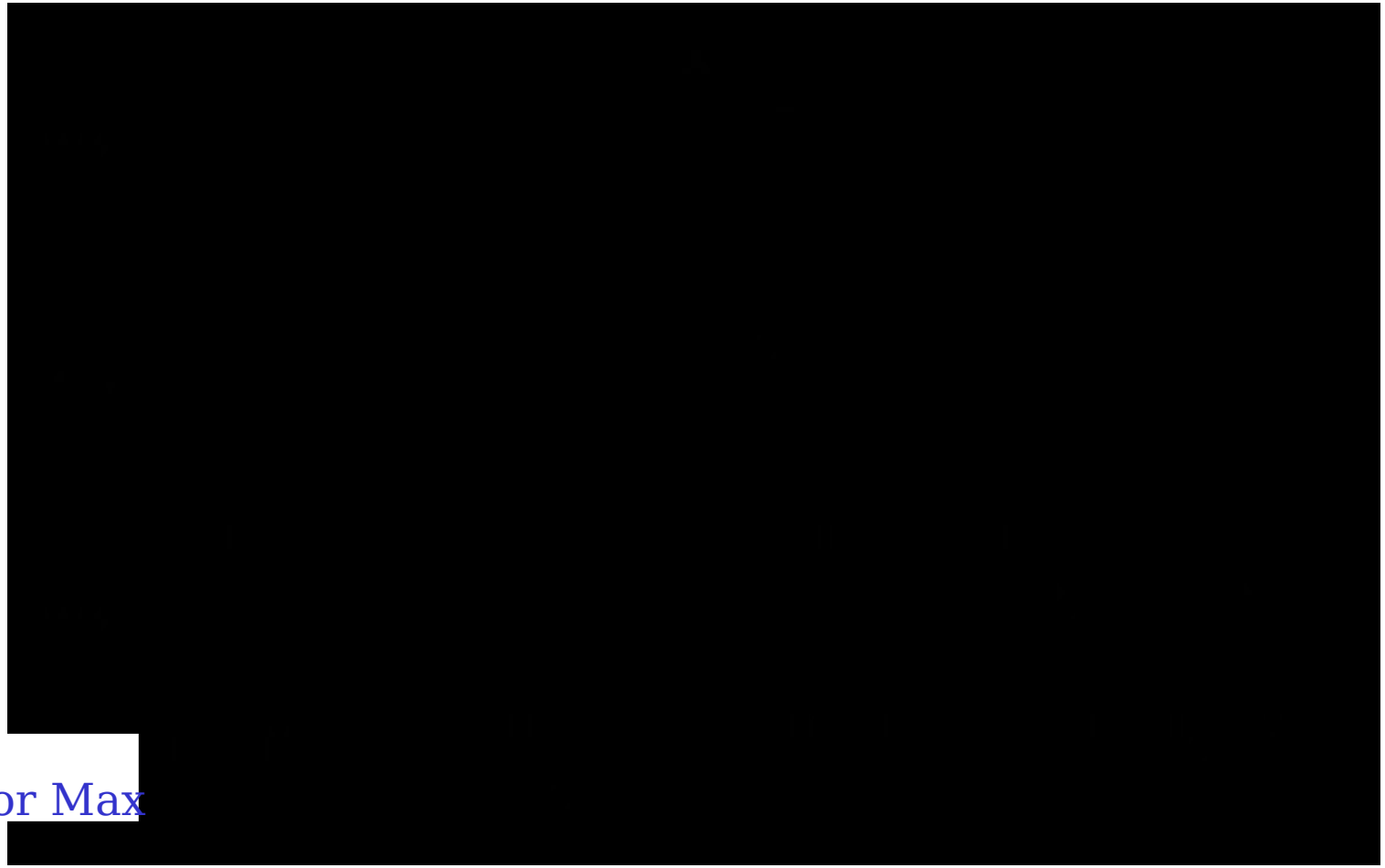
  ⇒ Identify best achievable payoff against best play

Minimax Algorithm



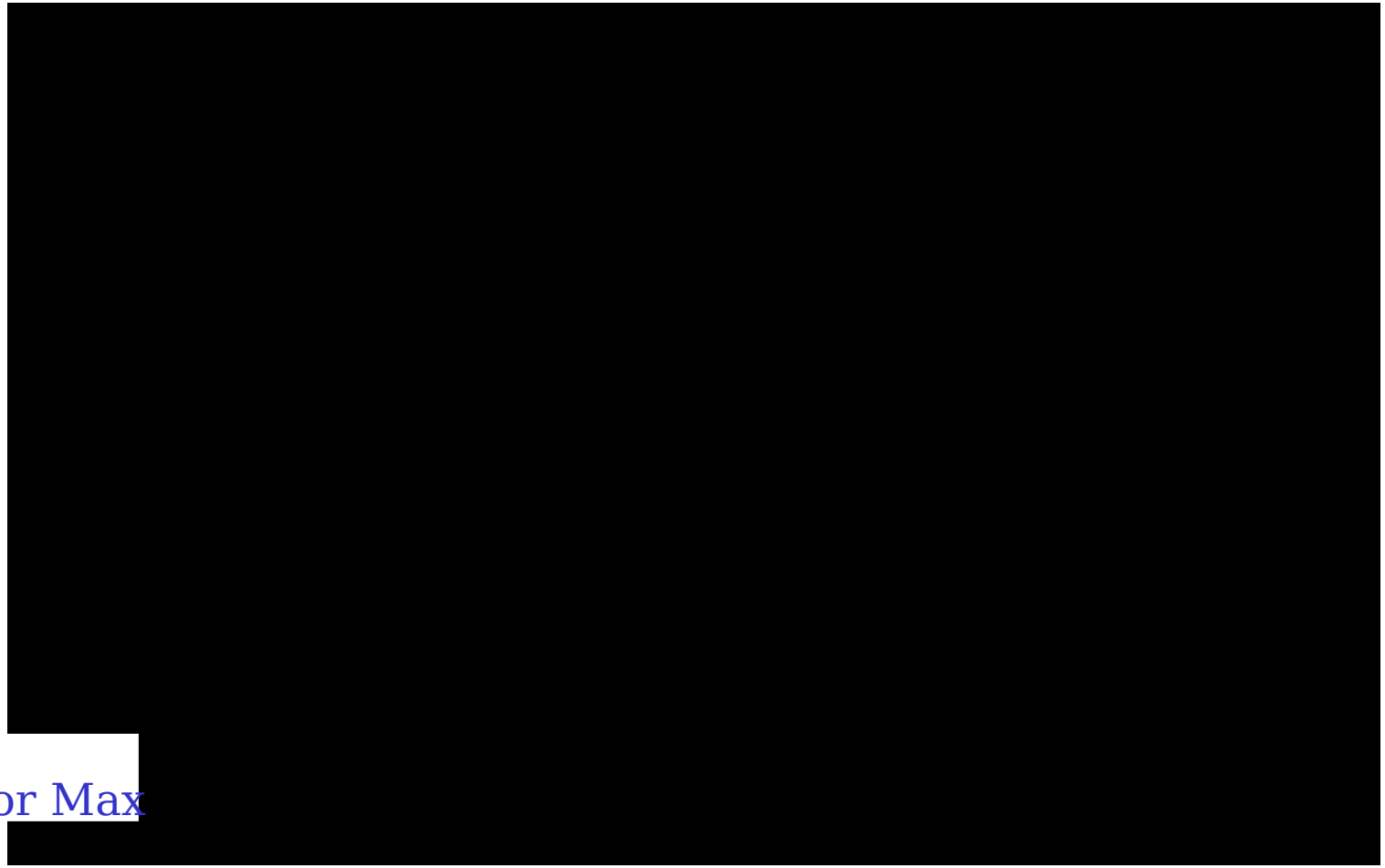
Payoff for Max

Minimax Algorithm (cont'd)



Payoff for Max

Minimax Algorithm (cont'd)



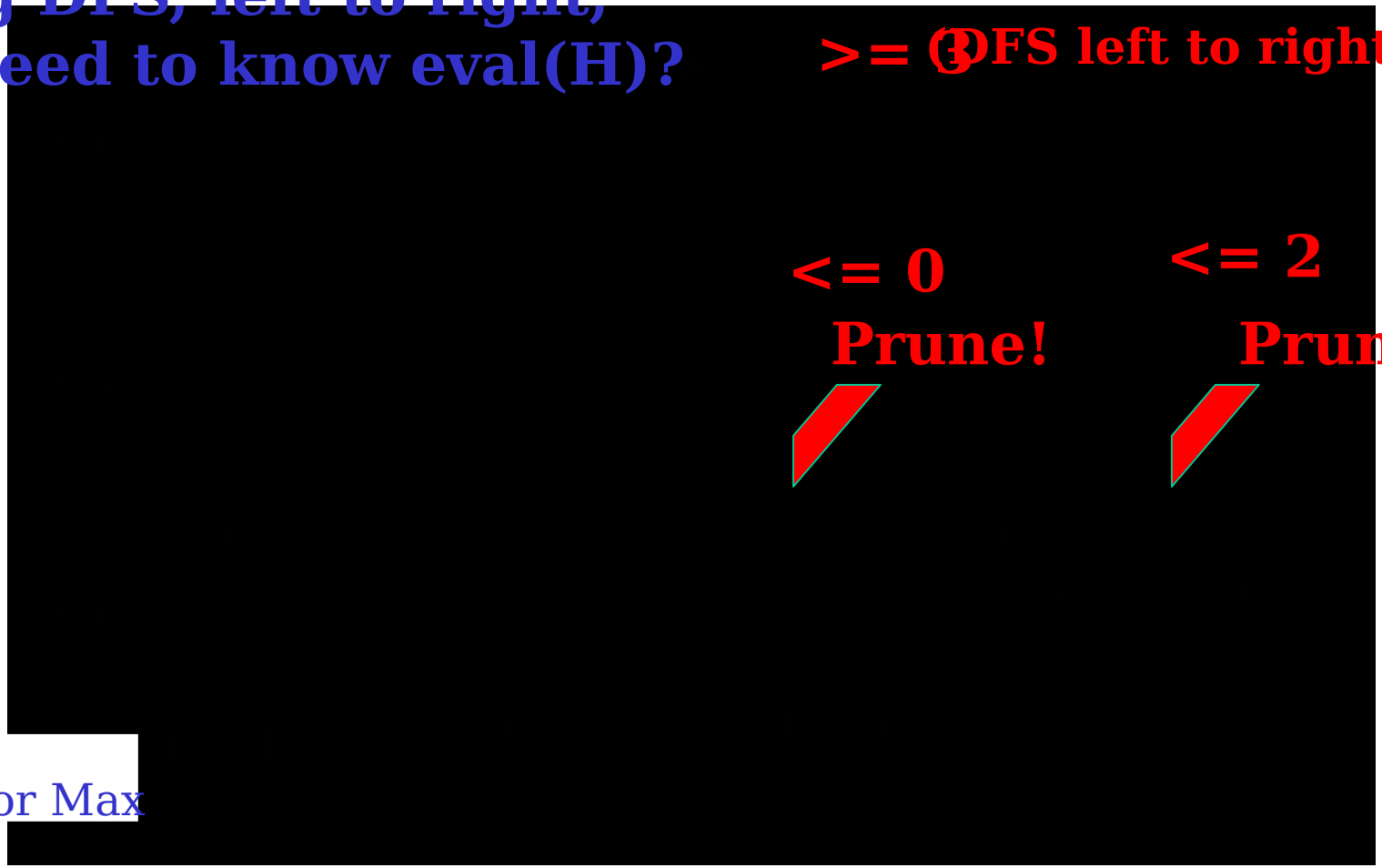
Payoff for Max

Minimax Algorithm

What if
payoff(Q) = 100
payoff(R) = 200

Do DFS. Real games: use iterative deepening.
(gives "anytime" approach.)

Starting DFS, left to right, do we need to know eval(H)?



Payoff for Max

Properties of minimax algorithm:

Complete? Yes (if tree is finite)

Optimal? Yes (against an optimal opponent)

Time complexity? $O(b^m)$

Space complexity? O(bm) (depth-first exploration, if it generates all successors at once)

For chess, b ≈ 35, m ≈ 80 for "reasonable" games
⇒ exact solution completely infeasible

m – maximum depth of the tree; b – legal moves

Minimax Algorithm

Limitations

- **Generally not feasible to traverse entire tree**
- **Time limitations**

Key Improvements

- **Use evaluation function instead of utility (discussed earlier)**
  - **Evaluation function provides estimate of utility at given position**
- **Alpha/beta pruning**